7N-18-TM
91481
P-11

Interactive Remote Control for an STS-Based Superfluid Helium Transfer Demonstration

Jeff Shapiro Frank Robinson

Technical Report FIA-89-10-03-02
October 1989

NASA Ames Research Center

Artificial Intelligence Research Branch

Mail Stop 244-17
Moffett Field, CA 94035-1000

INTERACTIVE REMOTE CONTROL FOR AN STS-BASED SUPERFLUID HELIUM TRANSFER DEMONSTRATION

Jeff C. Shapiro
&
Frank A. Robinson
Sterling Software

Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 244-17
Moffett Field, CA 94035

Abstract

The Superfluid Helium On-Orbit Transfer (SHOOT) experiment is a Shuttle-based demonstration of the technology required to service cryogenically cooled satellites in space. The experiment will be controlled continuously during a four to seven day mission by a scientist at a Payload Operations and Control Center (POCC).

The SHOOT Command and Monitoring System (CMS) software, developed on a Macintosh II, will provide a near real time highly interactive interface for the scientist to control the experiment and to analyze and display its telemetry. The CMS software will communicate via the NASCOM network with the experiment located in the space shuttle's cargo bay. Important features of the CMS include a graphical point-and-click interface making the software very easy to use with only a limited knowledge of the experiment. The design of the software was driven by the user interface which was prototyped completely before implementation, and is general enough to be reusable throughout all phases of the experiment. SHOOT is being managed by Goddard Space Flight Center and the software is being developed at Ames Research Center.

Introduction

The Superfluid Helium On-Orbit Transfer (SHOOT) experiment is a demonstration of the critical technology required to service cryogenically cooled satellites in a micro-gravity environment. Liquid helium is in the superfluid state at temperatures less than about 2 °K at normal atmospheric pressures. Superfluid helium has a number of unusual properties including zero viscosity, very high thermal conductivity, and a thermo-mechanical effect in which the helium is attracted to heat.

During the mission, a scientist at a Payload Operations and Control Center (POCC) will use the SHOOT Command and Monitoring System (CMS) to interactively operate the experiment from the ground. Running on a Macintosh II, the CMS software will allow the scientist to generate command packets to be up-linked to the experiment which will open and close valves, turn on heaters and pumps, and select the order and frequency of monitoring on-board sensors to be included in the telemetry. The telemetry is then analyzed and may be displayed in various formats. Important status information is always displayed graphically so the user can quickly see if something is wrong.

Experiment Objectives

During experiment operations, a number of transfers will occur in which helium is pumped from one dewar to another. Each dewar is heavily instrumented with temperature and pressure sensors, liquid/vapor detectors, and liquid level detectors, and can hold about 210 liters of liquid helium. Key components to be tested include thermo-mechanical pumps, the vacuum-jacketed transfer line, an EVA rated transfer line coupler, and the fluid acquisition system. A technique called heat-pulse mass gauging which very accurately determines the mass of helium in each dewar will also be verified.[1] While the SHOOT experiment will reside entirely aboard the space shuttle (Figure 1), the technology and control software developed may be readily applied to systems servicing free flying satellites requiring

This paper was presented at the AIAA Computers in Aerospace VII Conference, October 3-5, 1989, Monterey, CA.

cryogen resupply. One such proposed system is NASA's Superfluid Helium Tanker which may be deployed from either the shuttle or space station.

Most of the time the experiment is controlled by a scientist at the POCC. However there are several operations which require real-time feed back that cannot be done on the ground due to transmission delays in the down-link and up-link. These operations involve a coordinated effort between the scientist (on the ground) and an astronaut monitoring the experiment with the Aft Flight Deck computer (a flight certified PC-compatible laptop running independent software) who will fire the shuttle's Reaction Control System (RCS) engines to create small g-forces. A single transfer operation performed solely by an astronaut is also planned to demonstrate autonomous control without intervention from the POCC system.[2]

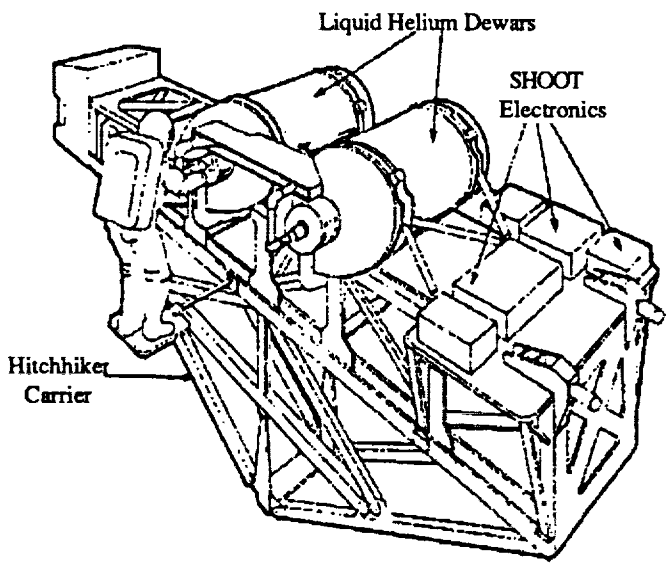


Figure 1. SHOOT/Hitchhiker Configuration

Design

Because the CMS software is used to control the experiment remotely, the scientist does not have direct visibility into the hardware. Therefore the CMS must give the user maximum flexibility while at the same time allowing operations to be performed quickly and easily. Telemetry must be presented in an intuitive format since the state of the experiment will determine what operations will occur next within a short period of time. It must also be available for display at the lowest level to assist in the diagnosis of error conditions.

Requirements

Ideally a firm set of software requirements are defined by the user and given to the software designer. Realistically, however, this was not the case and it was necessary to learn as much as possible about the experiment and define our own requirements.[3] These were then reviewed by the principal users and modified in an iterative process. Prototypes of the user interface were also reviewed in a similar fashion. Important requirements for the CMS include: redundancy to guarantee archiving of telemetry; real time calibration of raw data; display of data in a graphical format including real time plotting; full commanding capabilities including pre-defined macros (lists of commands), creation of new macros on-line, and automatic command generation for repetitive operations; and error checking to alert the user to an abnormal condition and to prevent invalid commanding (see Table 1).

- Support testing of SHOOT system, pad servicing and real time flight operations with identical software and functionally identical hardware.
- Archive converted and raw data during flight.
- Send commands to SHOOT electronics and archive command log.
- Plot sensor readings as a function of time.
- Support hardcopy output of screen images.
- Provide graphical user interface with mouse and menu control.
- Convert all data faster than real time.
- Provide utilities that aid user in real time decision making.
- Be single failure tolerant for experiment success.

Table 1. CMS General Requirements

Goals

The goal in designing the CMS software was to create a system that is easy to use and automates routine tasks while retaining full control of the experiment at a low level. Because the SHOOT experiment is highly interactive and will be controlled for extended periods of time, having a good user interface is very important. The Macintosh's icon based (Figure 2) point-and-click approach provides quick and easy methods for performing complicated tasks. The familiar windowing environment contributes to the ease of operation, decreasing the likelihood of errors made by the user. It is also non-modal so the user is never trapped by an action and every function is always available. The user interface of the CMS was fully prototyped before any code was written and drove the design of the software using a top down approach. A window was designed from the user's perspective to perform each required function. In Addition, the software conforms to the Macintosh User Interface Guidelines published by Apple Computer.[4]

Figure 2. CMS Icons

Flexibility will allow the software to be reusable for all aspects of the experiment including early tests with the hardware, system level integration, servicing the experiment in the payload bay of the shuttle while on the launch pad, and operations during and after flight.[5] Should the CMS become overloaded it is designed to degrade gracefully. Every function in the system has a certain priority; lower priority operations will be gradually phased out until higher priority operations have time to catch up.

Implementation

The CMS software is being developed on a Macintosh II using the Macintosh Programmers Workshop (MPW) and is being written in C. The event oriented operating system on the Macintosh enables simultaneous user interaction and internal processing to occur without having multiple processes. The software is logically broken down into modules corresponding to functionality (Figure 3). Priority is highest for archiving and calibrating telemetry, followed by commanding operations and updating the display.

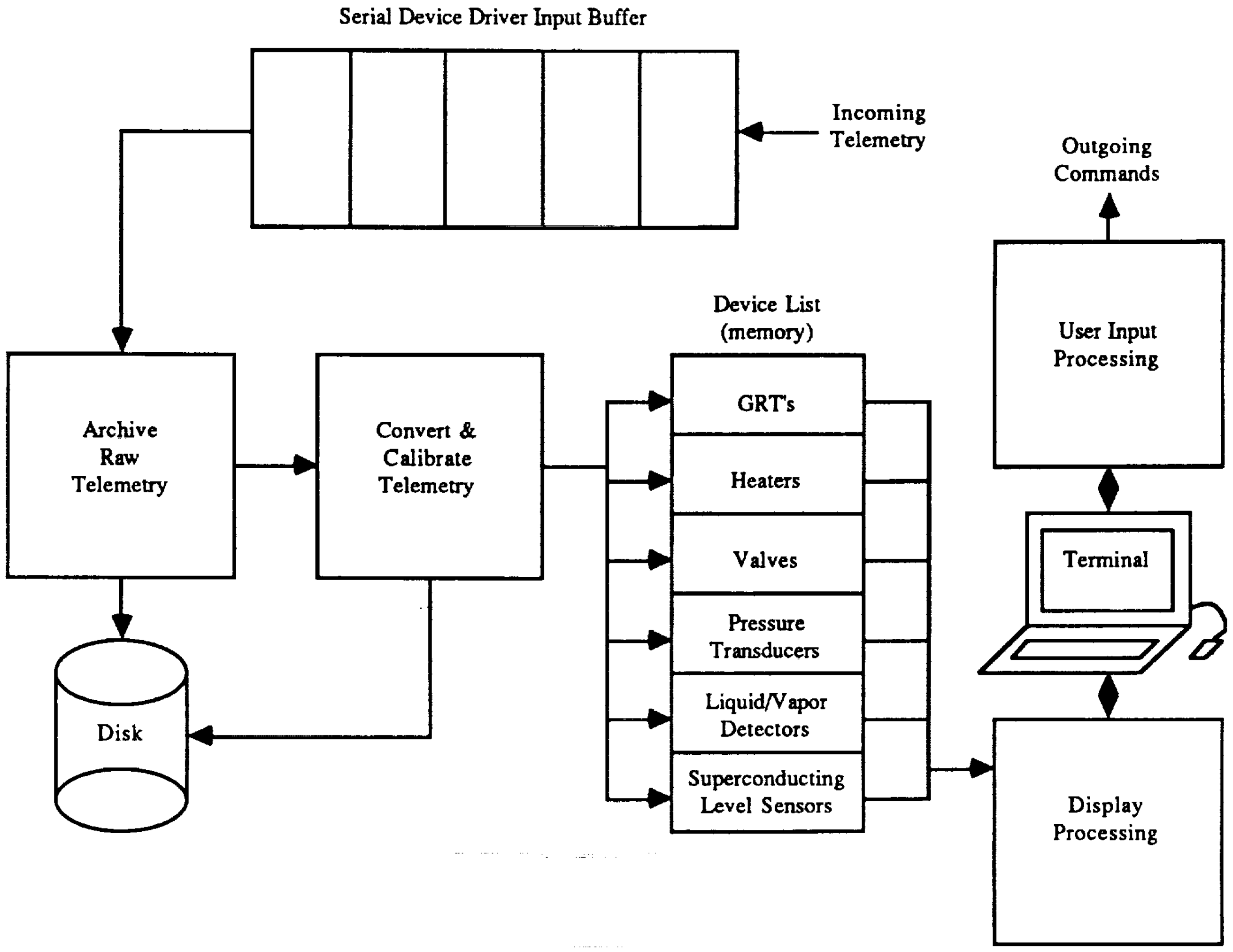


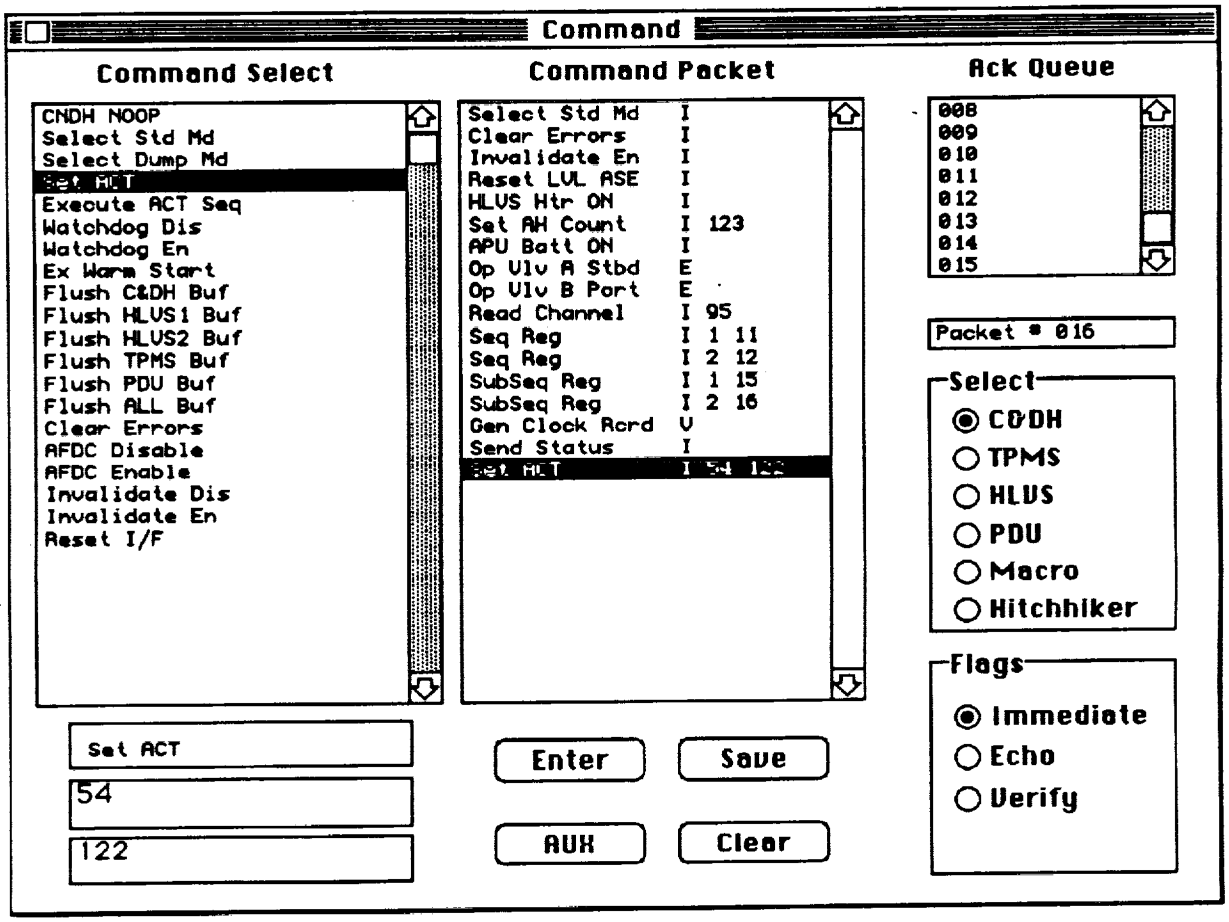
Figure 3. CMS Functional Diagram

User Interface

Telemetry processing consists of buffering the data as it comes in through the serial port, decommutating a data packet, performing conversions of raw data into engineering units, and archiving both raw and converted data to disk. Most of this occurs invisibly to the user, as described later. All telemetry is archived raw; however the user may select which devices should be converted, limit checked, displayed in a scrolling text window, and generate alerts if out of limits. The user will always be notified if any telemetry indicates an error condition. A debug window exists for displaying raw telemetry packets which is useful during pre-flight testing but is not practical during actual flight operations.

Command processing is provided by a portion of the user interface to construct command packets and a command assembler which formats the packet for up-link and sends it out the serial port. The command window (Figure 4) has lists broken down by SHOOT subsystem containing every command. This makes it very difficult for the user to make an invalid entry into a command packet. An auxiliary feature allows entry of a command that is not pre-defined in one of the lists, but this can be disabled during flight. Parameters are entered using the keyboard in familiar units which are converted and limit checked before being entered into the packet. If a command requires parameters it will not be put into the packet until they are entered. Parameters may not be entered for commands which have none. A macro facility exists to simplify the creation of a command packet. Pre-defined macros of commands (with necessary parameters) are available in a separate list. Macros may be of arbitrary length but may not be nested. A manually created packet may be saved as a new macro and is added to the list. Since this is also how all pre-defined macros are created, they are guaranteed to be valid.

To prevent sending a packet by accident, a modal dialog box requires verification by the user. Command packets may also be sent automatically for operations that need to be repeated a number of times. A queue is maintained of packets which have been up-linked but not yet acknowledged by the experiment. Packets are popped off the top of the



Command Select

```
CNDH NOOP
Select Std Md
Select Dump Md
Set ACT
Execute ACT Seq
Watchdog Dis
Watchdog En
Ex Warm Start
Flush C&DH Buf
Flush HLVS1 Buf
Flush HLVS2 Buf
Flush TPMS Buf
Flush PDU Buf
Flush ALL Buf
Clear Errors
AFDC Disable
AFDC Enable
Invalidate Dis
Invalidate En
Reset I/F
```

```
Set ACT
54
122
```

Command Packet

```
Select Std Md    I
Clear Errors     I
Invalidate En    I
Reset LVL ASE    I
HLVS Htr ON      I
Set AH Count     I  123
APU Batt ON      I
Op Vlv A Stbd    E
Op Vlv B Port    E
Read Channel     I  95
Seq Reg          I  1  11
Seq Reg          I  2  12
SubSeq Reg       I  1  15
SubSeq Reg       I  2  16
Gen Clock Rcrd   V
Send Status      I
Set ACT          I  54  122
```

Ack Queue

```
008
009
010
011
012
013
014
015
```

Packet # 016

Select: [✓ C&DH] TPMS HLVS PDU Macro Hitchhiker

Flags: [✓ Immediate] Echo Verify

Figure 4. Commanding Interface

queue when an acknowledgement is received and the packet is then archived. This queue is visible to the user so he can see how long the delay is between sending the packet and receiving the acknowledgement (estimated at several seconds to a minute). If a negative acknowledgement is received (indicating a transmission error in the up-link), or if no acknowledgement is ever received, a packet may be manually deleted from the queue and re-sent if desired. It may also be edited before re-sending.

Display processing is responsible for updating the screen and for changing display attributes. The telemetry being displayed is disjoint from telemetry processing in that old data may be viewed even if it is not present in the current down-link packet and all new data coming in is not necessarily being displayed, although it is still being checked for error conditions. The display for a specific device is updated as new corresponding data comes in. Each device may be individually selected for numerical display (in either raw or converted formats) and can be positioned to suit the user's preference. A subset of all devices is always displayed graphically. Status information that is derived from telemetry is also displayed in either a text format or a graphical summary. Alerts and other error conditions are made known to the user in both the telemetry display window and a separate alert window; in addition an audible bell may sound if desired.

The dewar graphics window (Figure 5) contains the most interesting information the scientist will normally want to see: valves, liquid/vapor detectors, liquid level detectors, temperature sensors, flow rate, and the mass in each dewar. Each class of device is drawn in a separate overlay which the user may select from a menu; thus only what is currently of interest need be displayed to prevent overcrowding. Some devices, such as valves, are found in every telemetry packet and will be updated continuously. Others, such as temperature sensors, only come down occasionally and may only be displayed if out-of-limits.

Command & Telemetry Processing

Internally, the functional decomposition of the CMS is intended to isolate the user interface from the mechanics of obtaining telemetry data, processing the data, updating system status, formatting and transmitting commands, and maintaining the history files. This was accomplished in part by defining two sets of global tables; one set for telemetry processing and one set for commanding.

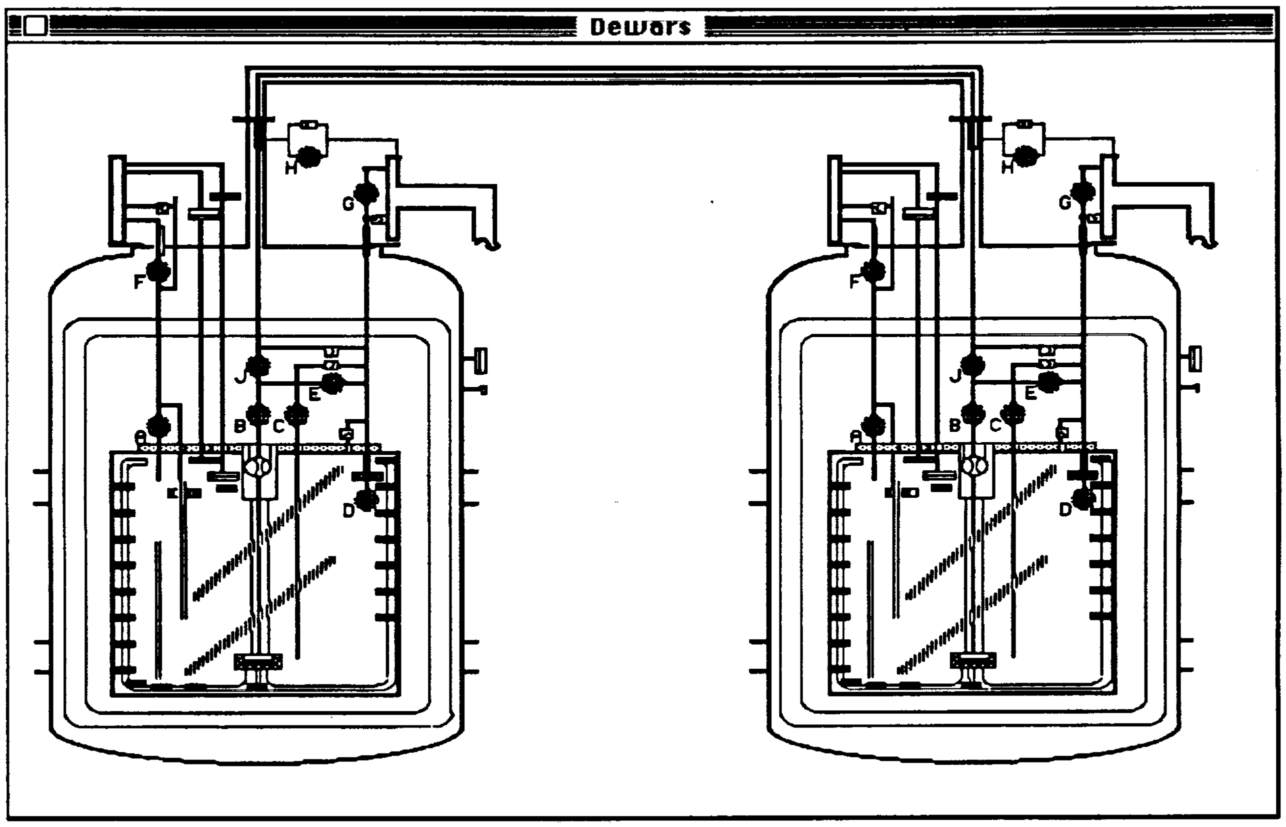


Figure 5. Graphical Telemetry Display

The telemetry processing tables are partitioned such that the data itself is contained in tables separate from the data descriptions which are used to decommutate the telemetry frame and perform the conversions. These tables are the Decommutation List, which relates each device to a location in the raw telemetry frame and to an entry in the device list; the Device List, which describes the required processing for that device, and which is dynamically modifiable in the sense of display, limits, and other dynamically selectable actions; and the Current Value Tables (CVT) which contain the most recent values, both raw and converted, for each measurand defined in the telemetry frame. There are four CVTs (Figure 6). One set of CVTs contains the data from most of the hardware. Some of the data (from the TPMS, described later) is handled separately due to its format which contains its own channel address and thus is decommutated using one additional level of indirection. The converted value CVTs are archived to disk, as is the raw telemetry frame itself. When invoked (i.e. valid telemetry has been received), the telemetry processor cycles through the lists and performs the activities specified in them. These tables are initialized from user configurable files at program startup.

The purpose of archiving the raw telemetry frame as a set of unformatted records (aside from record keeping) is to allow for a playback mode in which the CMS software is logically unmodified, and the raw data is read from files rather than from the serial port. The data descriptions may be altered in the user startup files in order to look at the data from different views. The purpose of archiving the processed data is to permit plotting during run time without having to recalculate values for previous data.

The commanding table is a straightforward lookup table consisting of an array of ascii strings (the command names) in parallel with the four byte command value, with command modifier information also in parallel. When the command assembler is invoked, this table is searched for a match to the string (command name) passed; the equivalent command value is placed in the command buffer. The system allows for as many as 28 commands to be stacked in a single pass. Command modifiers apply to those commands which incorporate address or magnitude information. These modifiers are OR'ed into the commands in the locations specified by the information contained in the commanding table. The command buffer is then output via

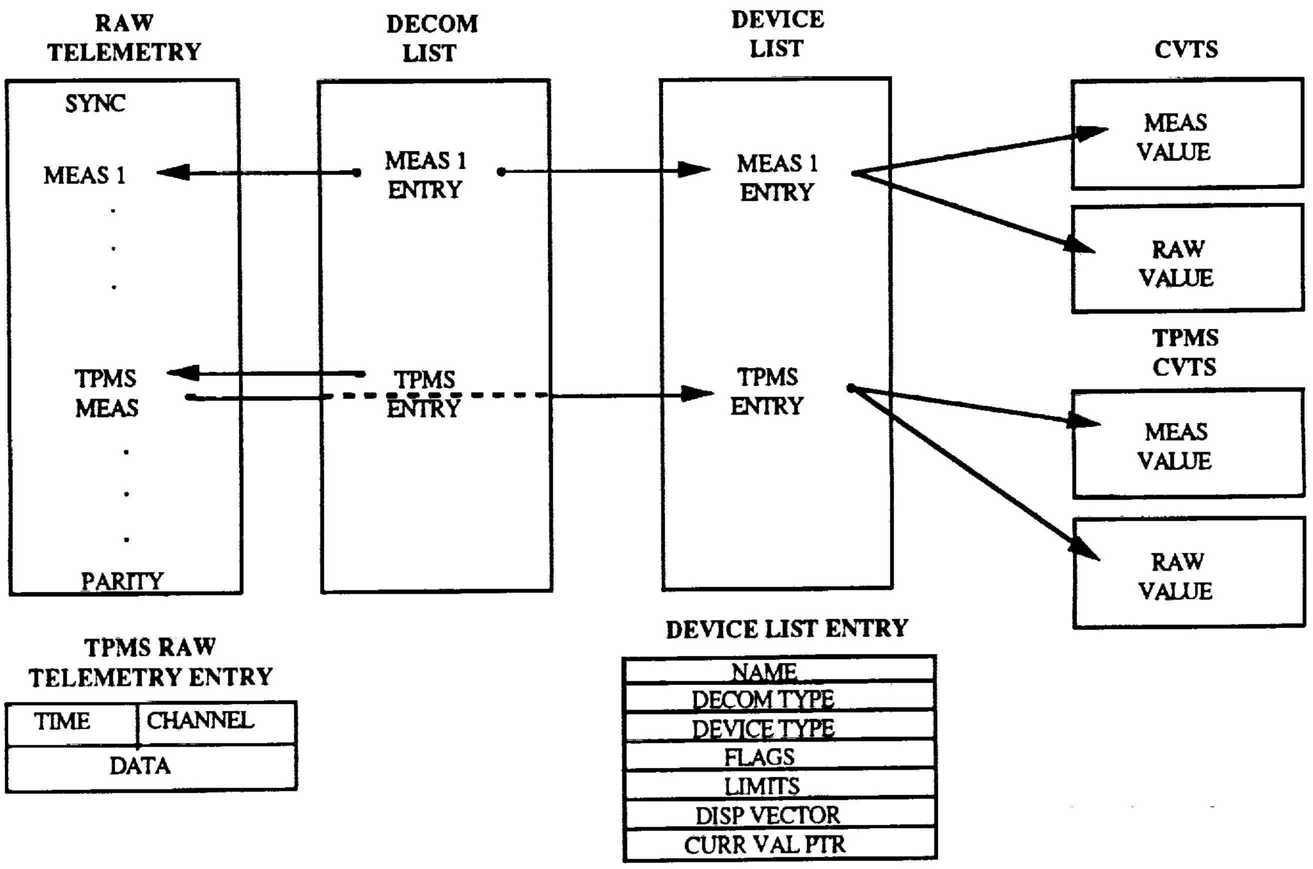


Figure 6. Telemetry Processing Lists

the serial port. On receipt of a command acknowledge in the telemetry stream, the ascii command array is archived.

Also generated at startup are arrays of measurand and command names, taken from the processing tables and placed in lists from which the user may select items for activation. Since this is the only mechanism by which the user can select a measurand for display (or a command for transmission), a direct correspondence is maintained between the user view of the system and the internal processing; the user cannot select an invalid option.

Conversion Functions

An important requirement of the CMS software is to convert raw device readings from on-board sensors into meaningful engineering units with which the scientist is familiar. This must be done in real time to keep up with the telemetry as it comes in. While the data rate is relatively slow (1200 baud nominally), some of the conversions are quite complicated. For example the raw data from a Germanium Resistance Thermometer (GRT) is a 16-bit digital count representing resistance. Obtaining a calibrated temperature from the raw count (maintaining full 16-bit accuracy) requires evaluating several 2nd order polynomials followed by two cubic-spline interpolations.[3] Of course many conversions are simply linear interpolations or table lookups.

Fortunately the 68881 floating point co-processor in the Macintosh II coupled with the Standard Apple Numerics Environment (SANE) is very fast. Initial benchmarking showed that 32 GRT readings (more than can fit into one telemetry packet) could be converted in about 1/5 of a second. Typically the most overhead is in updating the display so care was taken to re-draw only when necessary.

Utilities & Advisors

In addition to command and telemetry processing, several features will assist the user in operating the experiment. A plotting utility will allow old data which has already been archived, or new data as it comes in, to be graphed in a window. Multiple devices may be plotted versus time. This will enable the user to view selected devices and compare them; or see how they change over time.

One of the most important (and complicated) operations to be performed many times is mass gauging. During a mass gauging a known quantity of heat is injected into the helium and the resulting temperature differential plus the known heat capacitance give the mass. Temperature drifts, critical timing, and the operation of the mass gauge heaters must all be taken into account. The mass gauge advisor helps the user determine the optimum amount of heat to input into the dewar to obtain a large enough temperature differential without boiling off too much helium. An initial temperature and mass estimate are used to calculate the appropriate mass gauge heater voltage level and pulse duration. The voltage level and pulse duration must then be entered in the normal fashion in the command window. This was done to keep the commanding interface consistent. The mass gauge advisor monitors the telemetry and, after the heater has pulsed and enough drift information is obtained, calculates a new mass estimate. A similar advisor exists to calculate appropriate heater parameters to achieve a given flow rate (in liters/hour) during a transfer operation.

Hardware

Because the CMS software follows the guidelines published in Inside Macintosh, it is fully upward compatible with new Macintosh's and new versions of the operating system. This is desirable so that faster CPU's may be used if more speed becomes necessary. The CMS has already been tested successfully on the Macintosh IIx and IIcx, without any modifications, realizing a 20% improvement in performance. The software has also gone through several upgrades to the operating system and is compatible with MultiFinder.

Two identical systems will be used during flight at the POCC for redundancy, each capable of performing everything required to conduct the experiment, although only one system will be allowed to up-link commands at a time. The other system may be used for more time consuming tasks such as trend analysis of old data or plotting new data as it arrives in real time. Both computers will receive identical telemetry streams. Each system consists of a Macintosh II CPU with at least 5 MB RAM, 300 MB disk storage (which will probably increase) to hold archived telemetry, 19" and 13" color monitors, and a shared laser printer. Servicing the payload in the cargo bay while on the launch pad is required to top off the dewars with liquid helium as close as possible before launch to ensure an adequate supply in orbit. To perform pad servicing, two identical rack mounted Macintosh II's will be lifted next to the shuttle's payload bay (in the Payload Changeout Room) and connected directly to the experiment.

The POCC system communicates through a virtual serial connection with the experiment on board the orbiter. The Macintosh (located at a POCC at Goddard Space Flight Center) is connected to a micro-vax which is linked into the NASCOM network. Command packets are routed through Mission Control (at Johnson Space Center) to White Sands or a similar ground station where they are up-linked either directly to the shuttle or through a TDRSS satellite first. Once at the shuttle, the packet is routed through the Payload Signal Processor (PSP) to the Hitchhiker carrier on which SHOOT rides. The Hitchhiker then sends the packet to the SHOOT avionics.[6] Most of the data path is transparent, the CMS operates almost as if it is plugged directly into SHOOT (Figure 7). The only exceptions are a few commands which are intercepted by the Hitchhiker to supply power to the experiment. This is very advantageous because the same software used to test the experiment on the ground can be used during flight without modification. Telemetry is sent along the reverse path, except it is routed through the shuttle's Payload Data Interleaver (PDI).

The SHOOT electronics consist of several sub-systems connected to the dewars and instrumentation. Theses include the Command & Data Handling Unit (C&DH); the Temperature and Pressure Monitoring System (TPMS); the Heater, Level detector and Valve drive System (HLVS); and the Power Distribution Unit (PDU). The C&DH is responsible for processing command packets sent from the POCC and generating telemetry packets; and acts as the interface between the CMS software and the experiment hardware. The other sub-systems operate devices and read sensors.[1]

Conclusion

The development of the SHOOT Command and Monitoring System on a widely available and easy to use system, the Macintosh II, is proving to be both cost effective and efficient. The most important features are the point-and-click interface for interactive control and the reusable nature of the software. Operational tests with prototype electronics have already been performed successfully. The CMS software is being developed at NASA's Ames Research Center, primary responsibility for the SHOOT experiment is held by NASA's Goddard Space Flight Center. The target date for launch is late 1991 to early 1992.

Acknowledgements

The authors wish to thank the following for their support in the development of the SHOOT software: Tim Castellano (Sterling Software), Don Rosenthal (Ames Research Center), Dave Collins (Apple Computer), and Orlando Figueroa and the entire SHOOT team at Goddard Space Flight Center.

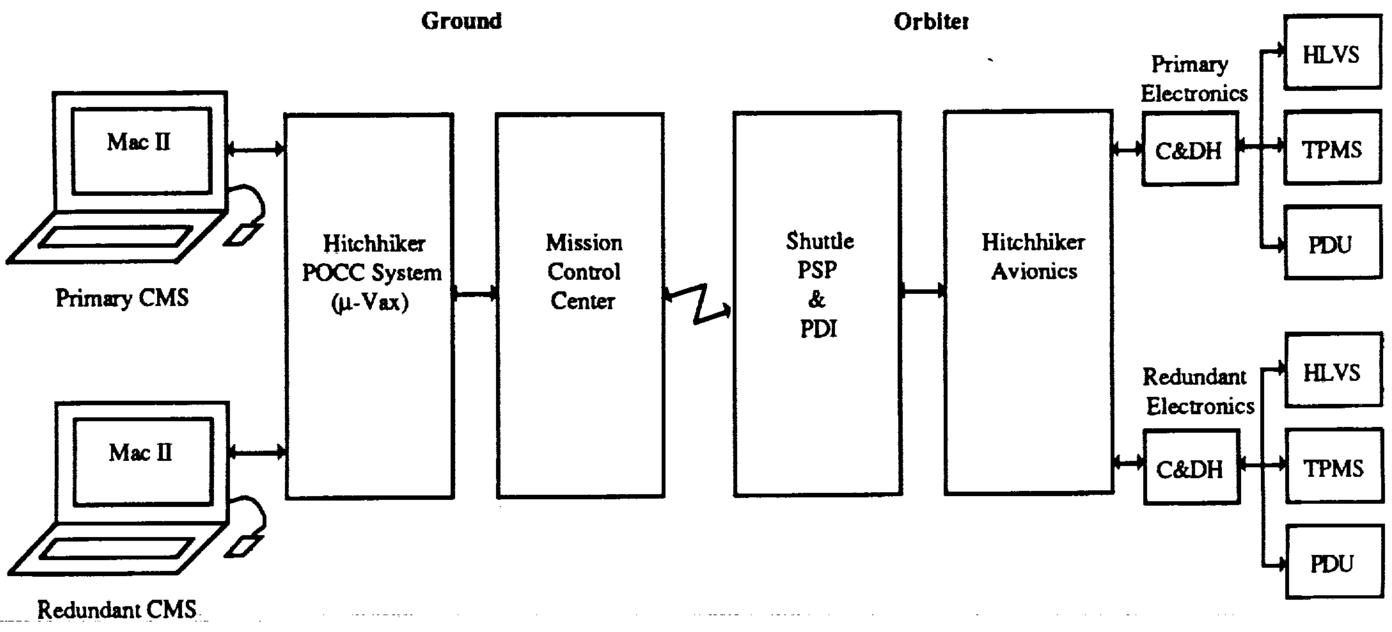


Figure 7. Hardware Configuration

References

[1] Woodhouse, C. E., Kashani, A., and Lukemire, A. T., "Superfluid Helium Tanker Instrumentation", IEEE IMTC, April, 1989.

[2] Castellano, T. P., Raymond, E. A., Shapiro, J. C., Robinson, F. A., and Rosenthal, D. A., "Knowledge Based and Interactive Control for the Superfluid Helium On-Orbit Transfer Project", 1989 Goddard Conference on Space Applications of Artificial Intelligence, NASA Conference Publication 3033, Greenbelt, Maryland, May 16-17, 1989.

[3] Castellano, T. P., Robinson, F. A., and Shapiro, J. C., "SHOOT Command and Monitoring System Software Requirements Document", NASA Publication SHOOT-SP-713-07, November, 1988.

[4] Apple Computer, "Inside Macintosh", Vol. I - V, Addison Wesley, 1985-1988.

[5] Kittel, P., and DiPirro, M. J., "Superfluid Helium On-Orbit Transfer (SHOOT) Operations", NASA Technical Memorandum 101009, July, 1988.

[6] Goddard Space Flight Center, "Hitchhiker Customer Accommodations and Requirements Specification", NASA Publication HHG-730-1503-04, July, 1988.

RIA-89-05-16-01

Knowledge Based and Interactive Control for the Superfluid Helium On-Orbit Transfer Project
Timothy Castellano, Eric Raymond, Jeff Shapiro, Frank Robinson, and Donald Rosenthal
May 1989

NASA's Superfluid Helium On-Orbit Transfer (SHOOT) project is a Shuttle-based experiment designed to acquire data on the properties of superfluid helium in micro-gravity. Aft Flight deck Computer Software for the SHOOT experiment is comprised of several monitoring programs which give the astronaut crew visibility into SHOOT systems and a rule based system which will provide process control, diagnosis and error recovery for a helium transfer without ground intervention, Given present Shuttle manifests, this software will become the first expert system to be used in space. The SHOOT Command and Monitoring System (CMS) software will provide a near real time highly interactive interface for the SHOOT principal investigator to control the experiment and to analyze and display its telemetry. The CMS software is targeted for all phases of the SHOOT project, hardware development, pre-flight pad servicing, in-flight operations, and post-flight data analysis.

RIA-89-10-03-01

Knowledge-Based Process Control and Diagnostics for Orbital Cryogen Transfer
Eric Raymond
October 1989

AFDeX is a rule based system designed to provide intelligent process control, diagnosis, and error recovery for a shuttle based cryogenic experiment, SHOOT (Superfluid Helium On-Orbit Transfer). This paper describes the AFDeX system in the context of traditional associative, model-based and qualitative systems, characterizes real-time features of the system, and discusses the implications of this first expert system in space.

RIA-89-10-03-02

Interactive Remote Control for an STS-Based Superfluid Helium Transfer Demonstration
Jeff Shapiro and Frank Robinson
October 1989

The Superfluid Helium On-Orbit Transfer (SHOOT) experiment is a Shuttle-based demonstration of the technology required to service cryogenically cooled satellites in space. The experiment will be controlled continuously during a four to seven day mission by a scientist at a Payload Operations and Control Center (POCC). The SHOOT Command and Monitoring System (CMS) software, developed on a macintosh II, will provide a near real time highly interactive interface for the scientist to control the experiment and to analyze and display its telemetry. The CMS software will communicate via the NASCOM network with the experiment located in the space shuttle's cargo bay. Important features of the CMS include a graphical point-and click interface making the software very easy to use with only a limited knowledge of the experiment. The design of the software was driven by the user interface which was prototyped completely before implementation, and is general enough to be reusable throughout all phases of the experiment. SHOOT is being managed by Goddard Space Flight Center and the software is being developed at Ames Research Center.

REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE Dates attached	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Titles/Authors - Attached	5. FUNDING NUMBERS
6. AUTHOR(S)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Code FIA - Artificial Intelligence Research Branch Information Sciences Division	8. PERFORMING ORGANIZATION REPORT NUMBER Attached
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Nasa/Ames Research Center Moffett Field, CA. 94035-1000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Available for Public Distribution [illegible] 5/14/92 BRANCH CHIEF	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstracts ATTACHED

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT